REMARKS/ARGUMENTS

Claims 1-3, 5-8, 13, 16-17, and 24-26 are pending in the present application. Claims 4, 9-12, 14-15, and 18-23 were canceled; claims 1-3, 5-6, 13, and 24 were amended; and claims 25-26 were added. Reconsideration of the claims is respectfully requested.

I. Interview Summary

The Examiner is thanked for the interview on June 9, 2006, at which the Examiner and the undersigned agent discussed both proposed amendments to the existing claims and new claim 25. It is understood from the discussion that the Examiner will be performing a further search of the art but that the features recited in the claims as amended overcome the current rejection.

II. 35 U.S.C. § 101

Claim 24 stands rejected under 35 U.S.C. § 101 as being directed towards non-statutory subject matter. This rejection is respectfully traversed.

Claim 24 has been amended to recite a "*computer readable tangible medium containing a computer program product*". The 101 rejection is therefore believed to be overcome.

III. Objection to Claims

Claim 19 stands objected to for a typographical error. Claim 19 has been cancelled; therefore, the objection is moot.

IV. 35 U.S.C. § 102, Anticipation

Claims 13, 16, 17 and 18 stand rejected under 35 U.S.C. § 102 as being anticipated by **Zimmer**, Using a Processor Cache as RAM During Platform Initialization, U.S. Patent No. 2004/0103272 A1, May 27, 2004 (hereinafter "**Zimmer**"). This rejection is respectfully traversed.

Claim 13 has been amended to better recite an embodiment of the invention; the amendments are supported by the application as follows:

- the system does not support instruction caching when reading from a first memory (page 5, line 19 through page 6, line 11);
- the first memory is organized into a plurality of pages (Figure 4, page 14, line 17 through page 15, line 7);
- the instructions are executed from a specific memory (Figure 5, page 15, line 8 through page 17, line 9) and

- the location of the second instructions affects when instruction caching is actually begun (Figure 6b, 618, 620, page 19, next to last line through page 21, line 17).

Amended claim 13 recites:

> 13. (Amended) An apparatus comprising:
> a processor,
> a first memory, which is organized as a plurality of pages, and a second memory, wherein the processor does not support instruction caching when executing from the first memory;
> initialization code including a first portion and a second portion, the initialization code having
> first instructions, executable from the first memory, for copying the first portion into the second memory,
> second instructions, executable from the first memory, for enabling instruction caching for the processor, wherein the second instructions are written at the end of a first page of the first memory and a second page following the first page is un-initialized, such that the processor automatically and temporarily disables instruction caching for the second page that is un-initialized,
> instructions, executable from the first memory, for transferring execution control to execute from the second memory, and
> instructions, executable from the second memory, for copying the second portion into a third memory.

A prior art reference anticipates the claimed invention under 35 U.S.C. § 102 only if every element of a claimed invention is identically shown in that single reference, arranged as they are in the claims. *In re Bond*, 910 F.2d 831, 832, 15 U.S.P.Q.2d 1566, 1567 (Fed. Cir. 1990). All limitations of the claimed invention must be considered when determining patentability. *In re Lowry*, 32 F.3d 1579, 1582, 32 U.S.P.Q.2d 1031, 1034 (Fed. Cir. 1994). Anticipation focuses on whether a claim reads on the product or process a prior art reference discloses, not on what the reference broadly teaches. *Kalman v. Kimberly-Clark Corp.*, 713 F.2d 760, 218 U.S.P.Q. 781 (Fed. Cir. 1983).

Zimmer does not anticipate the invention recited in claim 13 because **Zimmer** does not identically show every element of the invention as recited in this claim. For example, **Zimmer** does not disclose "*a first memory, ... wherein the processor does not support instruction caching when reading from the first memory*". Even more importantly, **Zimmer** does not disclose "*instructions, executable from the first memory, for enabling instruction caching for the processor, wherein the instructions for enabling instruction caching are at the end of a first page and the first page is followed by a second page that is un-initialized, such that the processor disables instruction caching for the second page that is un-initialized*". **Zimmer** discusses the initialization of an electronic device, but **Zimmer** does not address the management of instruction caching during the initialization process. Not all electronic devices need to address the issue of instruction caching during initialization. However, the present application notes at the bottom of page 5 that some systems inhibit instruction pre-fetching at certain times, such as initialization, by running in a cache-inhibited mode. The problem this can cause is illustrated on page 20 of the application when it notes that the firmware, operating in the inventive method, has "*prevented the hardware from being placed in a potential hang condition that could otherwise occur if a cache-inhibited device address were used in a cacheable instruction fetch operation.*

Since **Zimmer** does not disclose each and every feature of the invention recited in claim 1, **Zimmer** does not anticipate the claimed invention and this rejection is overcome.

Since claims 16-18 depend from claim 13, the same distinctions between **Zimmer** and the invention in claim 13 is applicable also to these claims. Consequently, it is respectfully urged that the rejection of claims 13 and 16-18 has been overcome.

Furthermore, **Zimmer** does not teach, suggest, or give any incentive to make the needed changes to reach the presently claimed invention. **Zimmer** does not discuss the enabling or disabling of instruction caching during the initialization process, nor does **Zimmer** provide a reason why this capability should be turned on or off at specific times. Absent the Examiner pointing out some teaching or incentive to implement the enabling or disabling of instruction caching during the initialization process of **Zimmer**, one of ordinary skill in the art would not be led to modify **Zimmer** to reach the present invention when the reference is examined as a whole. Absent some teaching, suggestion, or incentive to modify **Zimmer** in this manner, the presently claimed invention can be reached only through an improper use of hindsight using the applicants' disclosure as a template to make the necessary changes to reach the claimed invention.

V. <u>35 U.S.C. § 103, Obviousness</u>

Claims 1-8, and 19-24 stand rejected under 35 U.S.C. § 103 as being obvious over **Zimmer**, <u>Using a Processor Cache as RAM During Platform Initialization</u>, U.S. Patent No. 2004/0103272 A1, May 27, 2004 (hereinafter "**Zimmer**"), in view of Nalawadi, <u>Initialization with Caching</u>, U.S. Patent No. 2003/0023812 A1, January 30, 2003 (hereinafter "**Nalawadi**"). This rejection is respectfully traversed.

Claims 9-10 were rejected under 35 U.S.C. 103(a) as being obvious over **Zimmer** in view of **Nalawadi** and Dawson, <u>Dynamic Software Code Instrumentation Method and System</u>, U.S. Patent No. 2004/0025145, February 5, 2004 (hereinafter "**Dawson**").

Claims 11-12 were rejected under 35 U.S.C. 103(a) as being obvious over Avraham (US Patent Application Publication 2003/0233533), hereinafter "**Avraham**" in view of Springer et al (US Patent 6212631), hereinafter "**Springer**", further in view of **Dawson**.

Claims 14-15 stand rejected under 35 U.S.C. 103(a) as being obvious over **Zimmer**, **Avraham**, **Springer**, and **Dawson**.

It is noted that the subject matter of claims 9-10 has been incorporated into claim 1, the subject matter of claims 14-15 has been incorporated into claim 13, and claims 9-12 and 14-15 have been cancelled. Where subject matter of the cancelled claims is included in the remaining claims, their rejections will be discussed in reference to the independent claims.

Claim 1 has been amended to contain features similar to claim 13, discussed above. This claim now reads:

> 1. (Amended) A method of initializing an electronic device, comprising the steps of:
>
> in a system that does not support instruction caching when executing from a first memory, beginning execution of initialization code from the first memory, which is organized as a plurality of pages, to copy a first portion of the initialization code from the first memory into a second memory;
>
> while executing the initialization code from the first memory, performing a first instruction to software-enable instruction caching, wherein the first instruction is written at the end of a first page of the first memory and a second page following the first page is un-initialized, such that hardware on the electronic device will automatically and temporarily disable instruction caching for the second page that is un-initialized; and
>
> after execution of the first instruction, executing at least some of the first portion of initialization code from the second memory.

Zimmer appears to be the closest of the references relied on. As noted above, **Zimmer** does not address the issue of enabling instruction caching. The rejection asserts that **Zimmer** would have enabled instruction caching as originally claimed, noting:

> Zimmer et al do not explicitly mention that the I-Caching is software enabled, although, Zimmer et al mention that the cache locking is software enabled ([0014] of page 1) and the I-Cache may be enabled to initiate locking for dedicated use in initialization. The initialization is controlled by pre-boot software and it is likely that the cache is enabled by software, since the cache enabling is done in pre-boot time.

Office Action dated March 23, 2006, page 5, last paragraph

Applicants assert that **Zimmer** does not mention that instruction caching is software enabled because this application does not address instruction caching during initialization. Further, since not all systems must deal with this issue, it is not even clear whether **Zimmer** has any need to address this problem. Any assertion as to **Zimmer's** actions in this regard is not grounded in the reality of **Zimmer's** disclosure.

Additionally, the cache locking that is referenced in **Zimmer** is not the same action as enabling instruction caching. When instruction caching is enabled, the system is free to perform pre-fetching of instructions. On the other hand, when the cache is locked, the information already written in the cache cannot be overwritten. When given areas of the cache are locked, instruction caching can either be enabled or disabled; the two actions are not tied together. **Zimmer** notes that *"in the Intel XScale™ processor, up to 28 cache lines can be locked in a set. ... The code that performs the locking is cache inhibited. Instruction cache line fills cannot occur while the locking activity is in progress"*. **Zimmer** implies that the code is cache inhibited only during the time that the locking is taking place, as it would not be effective use of the processor if instruction caching was inhibited during the entire time that cache lines remain locked.

Nalawadi is cited as teaching "*a system where cache is enabled by software control ([0053] of page 4)*" (Office Action dated March 23, 2006, page 6, lines 1-2).

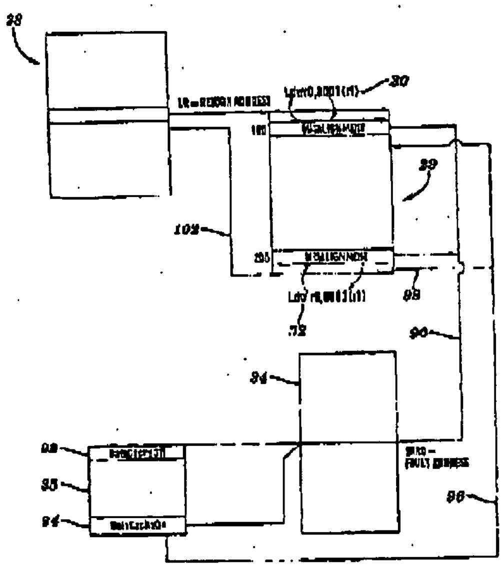

FIG. 2

Dawson is cited as teaching "*a system where cache enabling routine is located at the end of the page (94 of Fig 2)*". Figure 2 of **Dawson** is reproduced here and shows DataCacheOff **92** and DataCacheOn **94** as beginning and ending instructions for reference **35**. However, **Dawson** describes a portion of this figure thus:

> The exception thus generates a branch **90** to a conventional exception vector table **34**. An exception routine **35** is inserted into the vector table, which may include any of several instructions, such as disabling the Data Cache (as at **92**) and/or Instruction Cache. Subsequent instructions in the vector table execute the replaced preamble instruction and then, with or without re-enabling the cache as at **94** ...

Dawson, paragraph 34

As this text demonstrates, reference **35** does not represent a page in memory, but an exception routine that is stored in a conventional exception vector table **34**. **Dawson** does not show that the turning off of caching is performed at the end of a page, but at the end of a routine, whose physical location is not specified. Additionally, claim 1 further recites that the page following the cache-enable instruction is un-initialized and that the storage location of the cache-enable instruction provides a serendipitous advantage, in that "*hardware on the electronic device will automatically and temporarily disable instruction caching for the second page that is un-initialized*". Thus, it is the combination of the instruction, its location at the end of a page, and the following page being un-initialized that provides the ability to turn on instruction caching as early as possible, i.e., while operating from the first memory, while still avoiding the very real problem of hanging the system with such a command. None of the references relied on, either singly or in combination, suggest such an advantageous placement of this instruction.

The references relied on does not disclose, either separately or in combination, the feature "*in a system that does not support instruction caching when executing from a first memory, beginning* *execution of initialization code from the first memory*". Neither do these references disclose "*while executing the initialization code from the first memory, performing a first instruction to software-enable instruction caching, wherein the first instruction is written at the end of a first page of the first memory and a second page following the first page is un-initialized, such that hardware on said electronic device will automatically and temporarily disable instruction caching for the second page that is un-initialized*". Therefore, claim 1 is not obvious over these references. Additionally, claim 24 is rejected for the same reasons as claim 1 and its rejection is overcome for the same reasons. Additionally, claims 2-3 and 5-8 are dependent on claim 1, so their rejection is also overcome. Finally, claims 20-23 are dependent on claim 13, which now contains the same features as claim 1, so their rejection is also overcome.

Therefore, the rejection of claims 1-3, 5-8, and 20-23 under 35 U.S.C. § 103 has been overcome.

VI. New Claims

New claims 25 and 26 have been added to the application. Claim 25 is representative of claim 26 and recites:

> 25. A device comprising:
>
> a non-volatile memory containing initialization code, the non-volatile memory being organized as a plurality of pages, wherein the initialization code is stored in the non-volatile memory such that a first instruction to enable instruction caching is stored at the end of a first page in the non-volatile memory and the page following the first page is non-initialized;
>
> a cache memory;
>
> a system memory; and
>
> a processor, connected to the non-volatile memory, the cache memory and the system memory, wherein the processor hangs if instruction caching is attempted when executing from the first memory;
>
> wherein the initialization code is first executed from the non-volatile memory while a first portion of the initialization code is copied to the cache memory, the first portion of the initialization code is then executed from the cache memory while a second portion of the initialization code is copied to the system memory, and the second portion of the initialization code is finally executed from the system memory;
>
> wherein the storage location of the first instruction allows instruction caching to be software-enabled from the non-volatile memory but be hardware inhibited until execution passes from the non-volatile memory to the cache memory.

In this claim, the *wherein* clauses recite clear distinctions over the references relied on. None of the references discloses or suggests "*wherein the initialization code is stored in the non-volatile memory such that a first instruction to enable instruction caching is stored at the end of a first page in the non-volatile memory and the page following the first page is non-initialized*". Placing the first instruction at the end of a page that is followed by an un-initialized page is not a random choice, but a deliberate means of overcoming a hardware limitation in a very effective manner. Neither do the references disclose or suggest "*wherein the processor hangs if instruction caching is attempted when executing from the first* *memory*". This feature provides the motivation to place the first instruction in a deliberate manner that can optimize the time to initialize without causing the hardware to hang. Neither do the references disclose or suggest "*wherein the storage location of the first instruction allows instruction caching to be software-enabled from the non-volatile memory but be hardware inhibited until execution passes from the non-volatile memory to the cache memory*". None of the references give any suggestion that precise placement of elements of the initialization code can provide the recited benefit.

Therefore, these two new claims are believed to patentable over the cited references.

VII. Conclusion

It is respectfully urged that the subject application is patentable over the cited references and is now in condition for allowance.

The Examiner is invited to call the undersigned at the below-listed telephone number if in the opinion of the Examiner such a telephone conference would expedite or aid the prosecution and examination of this application.

DATE: June 23, 2006

Respectfully submitted,

Betty Formby
Reg. No. 36,536
Yee & Associates, P.C.
P.O. Box 802333
Dallas, TX 75380
(972) 385-8777
Agent for Applicant